体验流调试的威力 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/273/2021_2022__E4_BD_93_E9_AA_8C_E6_B5_81_E8_c104_273308.htm 如果你在使用流的过程中出现了问题，那么你也许需要调试功能。 Java的I/O框架是基于一系列连接在一起的流实现的。当这种设计增强了部件重用性的同时，它也使得定位错误变得困难。通过在流中插入你自己定义的调试流的时候，错误定位将会变得更简单。例如，调试流CountingOutputStream这个类报告它已经写了多少个字节。这个功能对使用flush造成字节丢失的情况非常有用。下面是一个使用CountingOutputStream的例子：package com.generationjava.io.import java.io.IOException.import java.io.OutputStream.import java.io.FilterOutputStream.public class CountingOutputStream extends FilterOutputStream {private int count.public CountingOutputStream(OutputStream out) {super(out).}public void write(byte[ ] b) throws IOException {count = b.length.super.write(b).}public void write(byte[ ] b, int off, int len) throws IOException {count = len.super.write(b, off, len).}public void write(int b) throws IOException {count = 2.super.write(b).}public int getCount( ) {return this.count.}} 在下面的小片代码中，存在一个简单的错误流既没有被flush也没有被关闭。File file = ...byte[ ] data = ...OutputStream out = new BufferedOutputStream(new FileOutputStream(file)).out.write(data). 插入CountingOutputStream来发现问题：File file = ...byte[ ] data = ...CountingOutputStream cos = new

CountingOutputStream(new FileOutputStream(file)).OutputStream out = new BufferedOutputStream(cos).out.write(data).System.err.println("Bytes written: " cos.getCount( )).如果这段代码输出：Bytes Written: 0那么可以很快的发现问题出在BufferedOutputStream上，它没有发送任何东西给FileOutputStream流。这种方法可以更容易发现作为缓冲的流没有flush它的缓冲。虽然这个例子过于简化了，但是它仍然证明了流调试的有效性。当链中存在四个或者更多的流的时候，插入并且移动调试流将帮助开发者更快的发现问题。